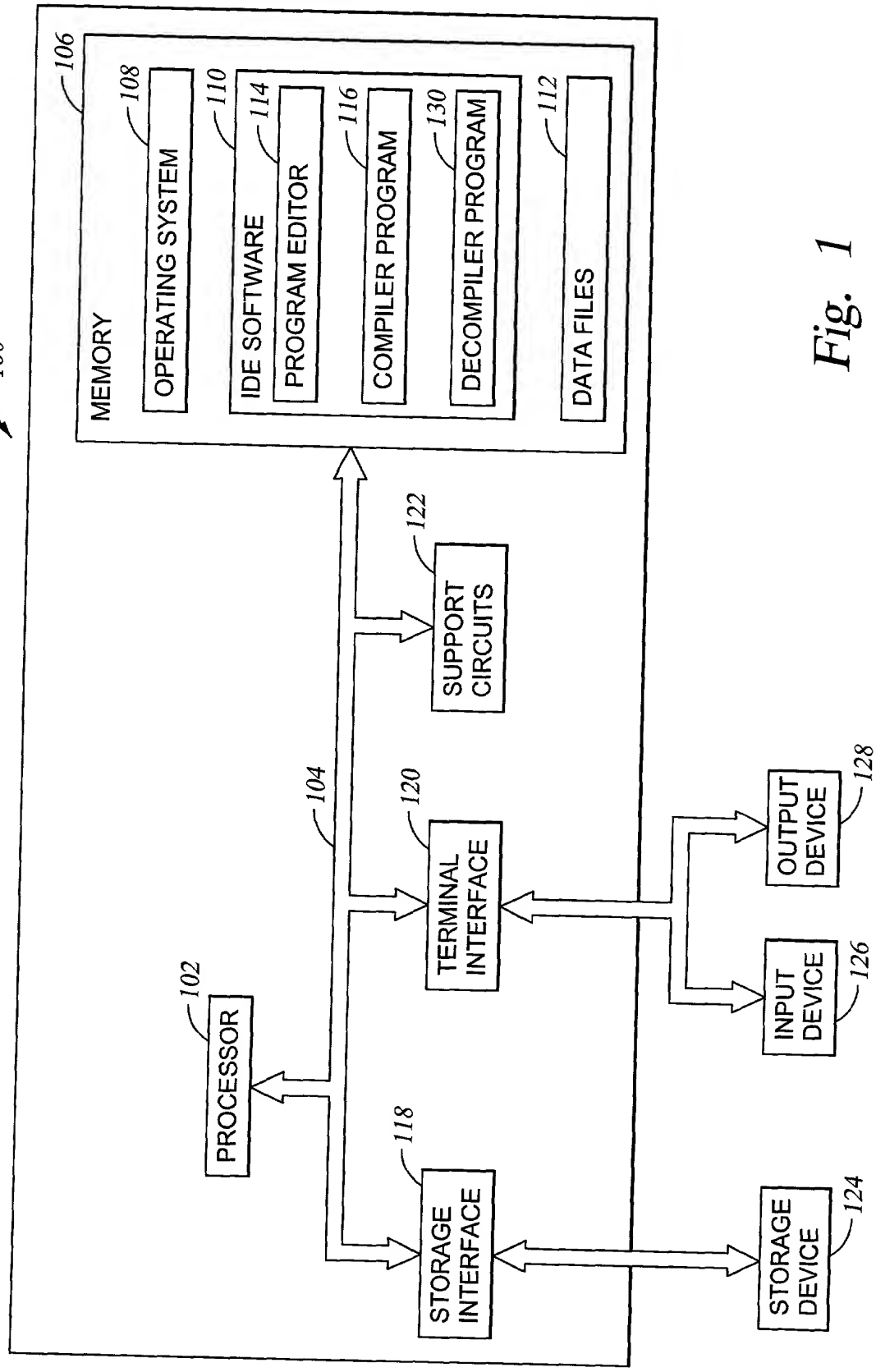

100
MEMORY
106
OPERATING SYSTEM
108
IDE SOFTWARE
110
PROGRAM EDITOR
114
COMPILER PROGRAM
116
DECOMPILER PROGRAM
130
DATA FILES
112
SUPPORT CIRCUITS
122
104
TERMINAL INTERFACE
120
OUTPUT DEVICE
128
INPUT DEVICE
126
PROCESSOR
102
STORAGE INTERFACE
118
STORAGE DEVICE
124

Fig. 1

Fig. 2A

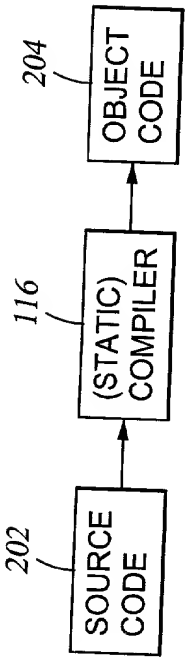


Fig. 2B

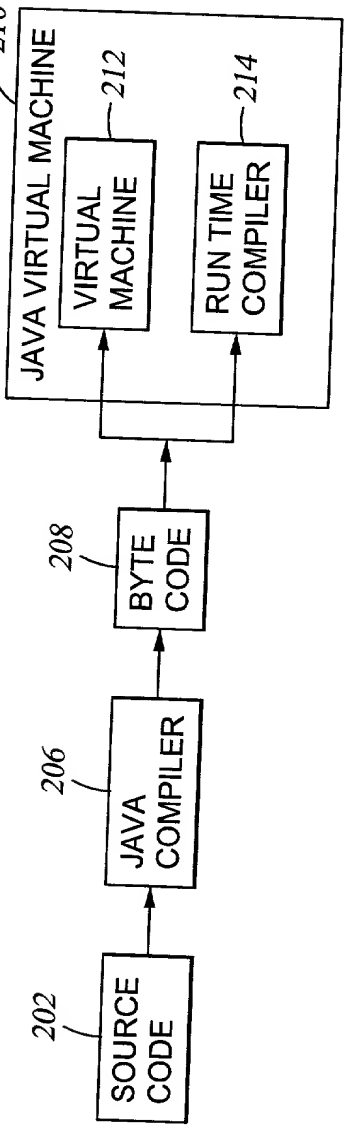
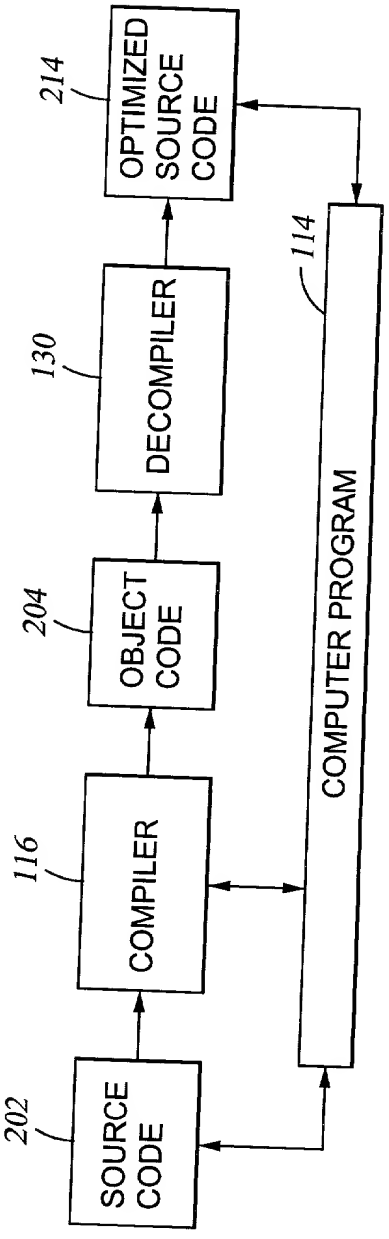


Fig. 2C



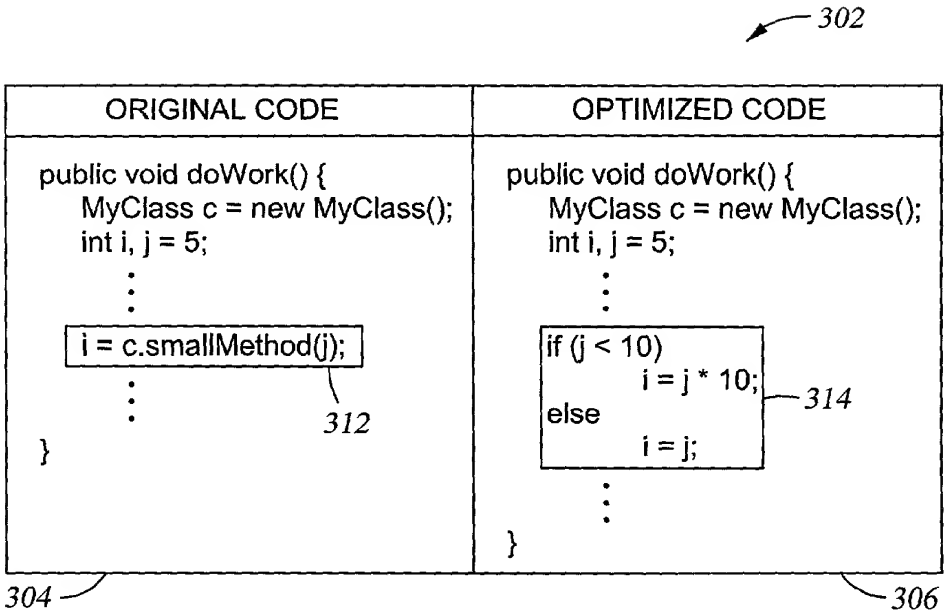

302
ORIGINAL CODE
OPTIMIZED CODE
public void doWork() {
MyClass c = new MyClass();
int i, j = 5;
i = c.smallMethod(j);
312
}
public void doWork() {
MyClass c = new MyClass();
int i, j = 5;
if (j < 10)
i = j * 10;
else
i = j;
314
}
304
306

Fig. 3A

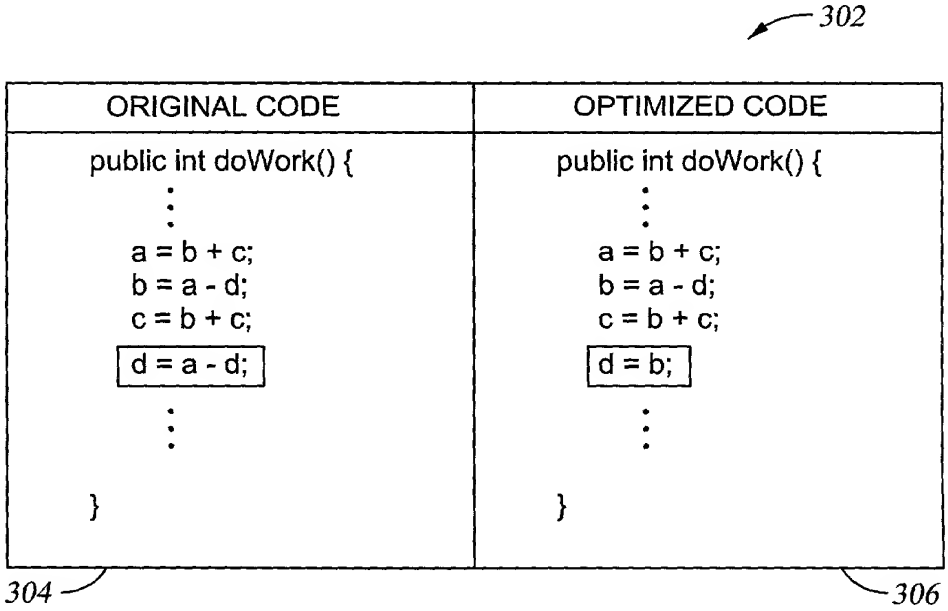

302
ORIGINAL CODE
OPTIMIZED CODE
public int doWork() {
a = b + c;
b = a - d;
c = b + c;
d = a - d;
}
public int doWork() {
a = b + c;
b = a - d;
c = b + c;
d = b;
}
304
306

Fig. 3B

302

ORIGINAL CODE	OPTIMIZED CODE

public int[ ] setIntVals() {
int[ ] intVals = new int[20] ;
for (int i = 0; i < intVals.length;
i ++) { intVals[i] = 1;
}
return (intVals) ;
 	
}
	public int[ ] setIntVals() {
	int[ ] intVals = new int[20] ;
	int len = intVals.length;
	for (int i = 0; i < len; i++) {
	intVals[i] = 1;
	}
	return (intVals) ;
	}

332 334

304 306

Fig. 3C

302

ORIGINAL CODE	OPTIMIZED CODE

public int[ ] setIntVals() {
int[ ] intVals = new int[20] ;
int len = intVals.length;
for (int i = 0; i < len; i++) {
intVals[i] = 1;
}
return (intVals) ;
}
	Not Applicable
	This optimization was not performed. While a loop was found in the method, all statements executed within the loop are dependent on results calculated during each iteration of the loop.

304 306

Fig. 3D

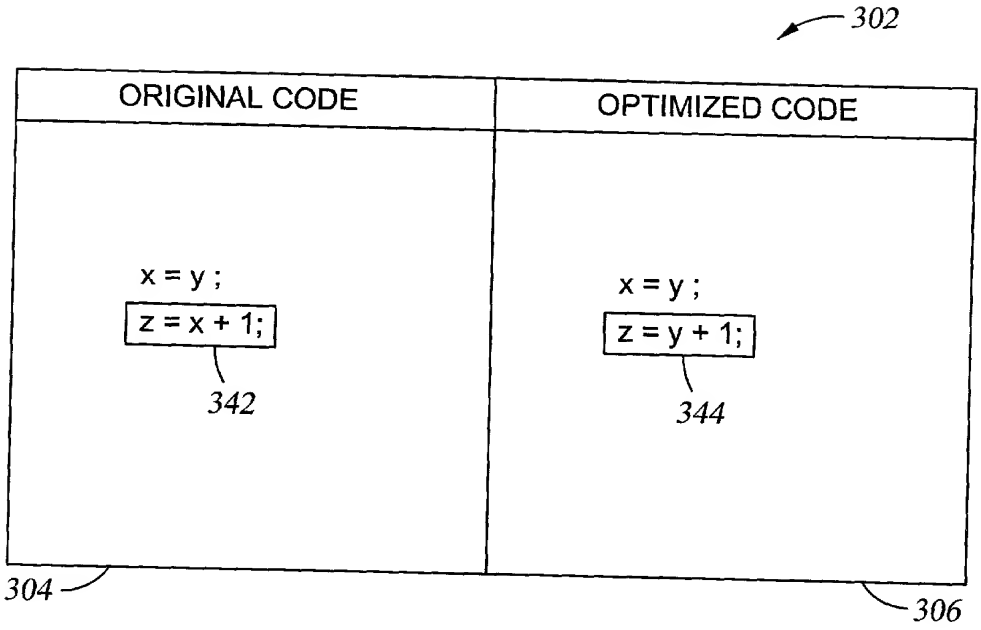

302
ORIGINAL CODE
OPTIMIZED CODE
x = y ;
z = x + 1;
342
x = y ;
z = y + 1;
344
304
306

Fig. 3E

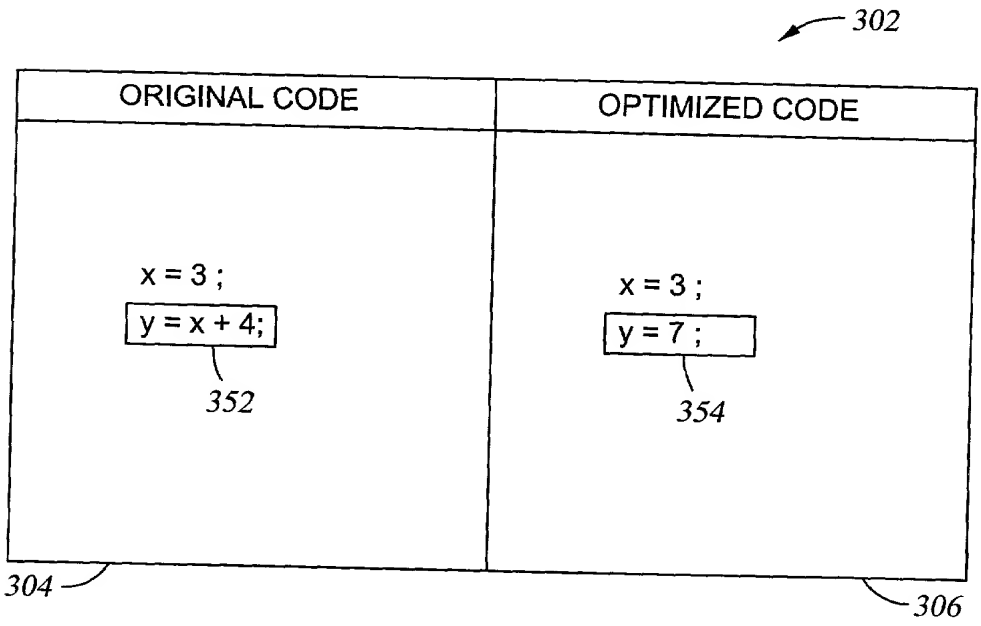

302
ORIGINAL CODE
OPTIMIZED CODE
x = 3 ;
y = x + 4;
352
x = 3 ;
y = 7 ;
354
304
306

Fig. 3F

302

" Just in Time " Compiler Heuristics

Times Procedure is Executed	Optimization
1 - 20	None (interpreted)
21 - 50	First stage optimization
51 - 100	Second stage optimization
101 +	Full optimization

Fig. 3G

302

Inlining Statistics for Project XYZ
53 methods from 25 types inlined in 134 places

Inlining Statistics for Package com.xyz.pkg1
12 methods from 7 internal types inlined in 44 places (intra-pkg)
7 methods from 2 external types inlined in 15 places (inter-pkg)

Inlining Statistics for Package com.xyz.pkg2
4 methods from 2 internal types inlined in 4 places (intra-pkg)
17 methods from 11 external types inlined in 31 places (inter-pkg)

Inlining Statistics for Class com.xyz.pkg1.ExampleClass
2 internal methods inlined in 4 places (intra-class)
3 external methods inlined in 6 places (inter-class)

Fig. 3H

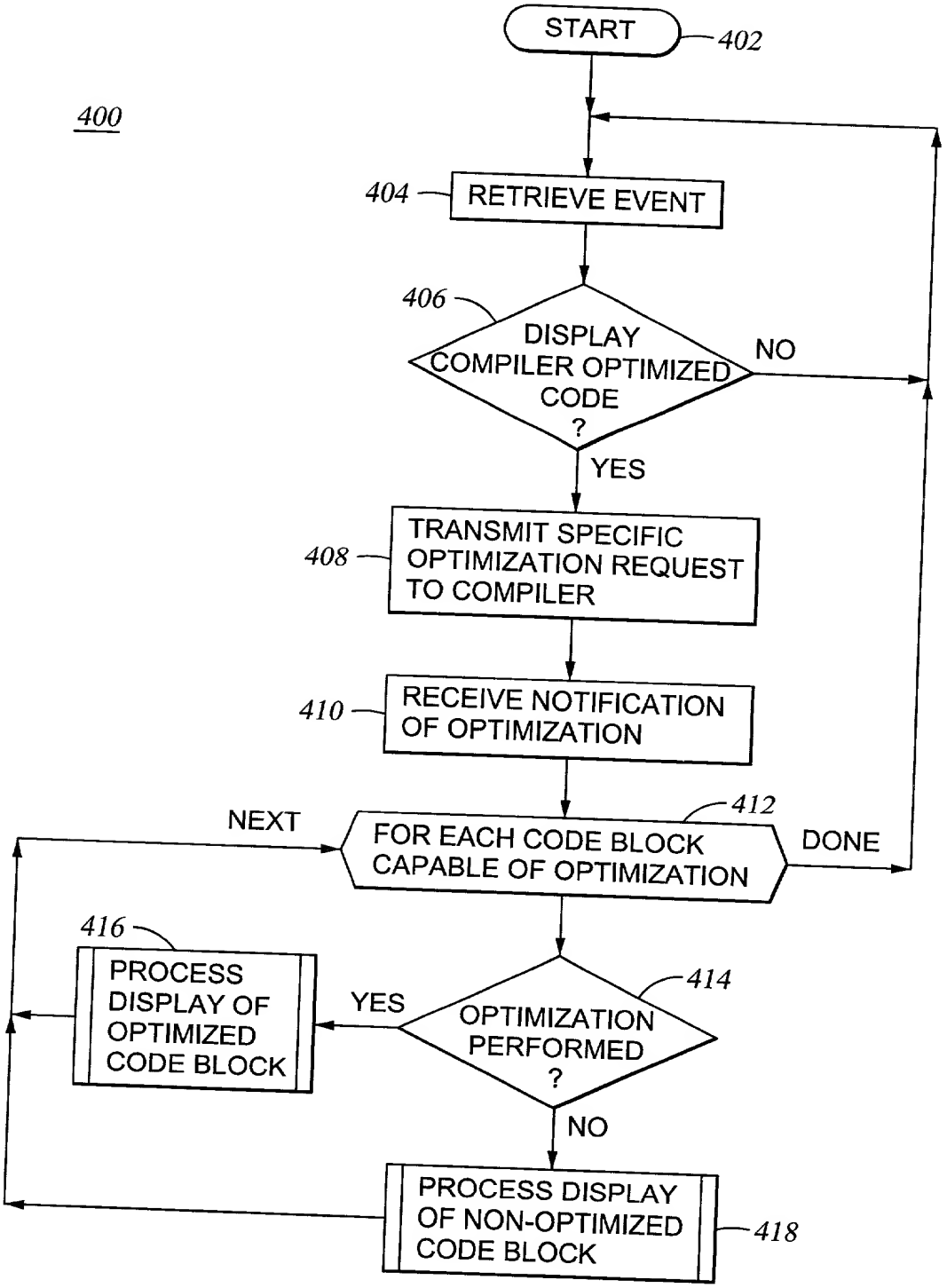

400
START
402
404
RETRIEVE EVENT
406
DISPLAY COMPILER OPTIMIZED CODE ?
NO
YES
408
TRANSMIT SPECIFIC OPTIMIZATION REQUEST TO COMPILER
410
RECEIVE NOTIFICATION OF OPTIMIZATION
412
NEXT
FOR EACH CODE BLOCK CAPABLE OF OPTIMIZATION
DONE
416
PROCESS DISPLAY OF OPTIMIZED CODE BLOCK
YES
414
OPTIMIZATION PERFORMED ?
NO
PROCESS DISPLAY OF NON-OPTIMIZED CODE BLOCK
418

Fig. 4

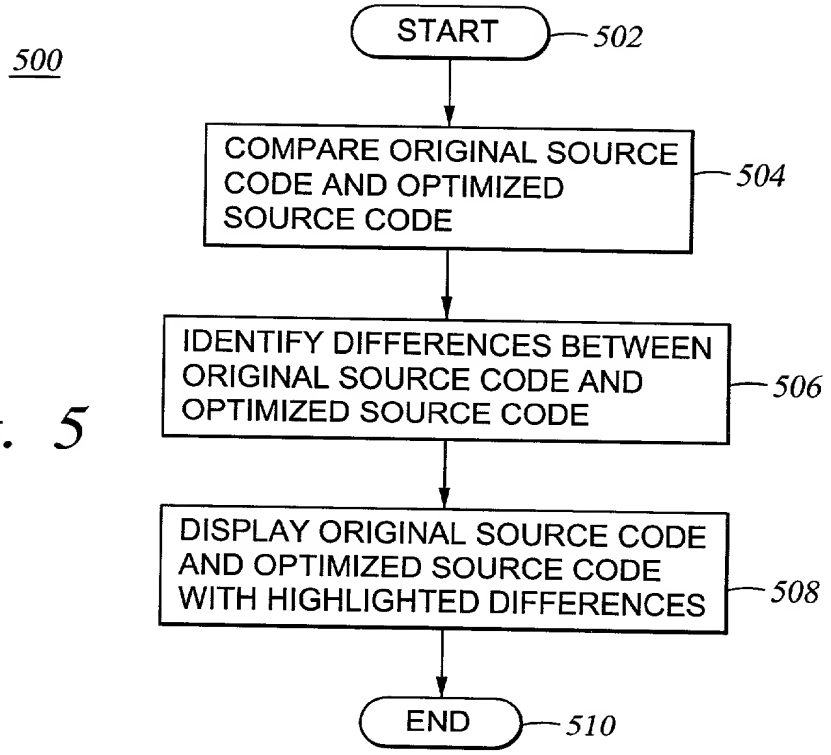

500
START
502
COMPARE ORIGINAL SOURCE CODE AND OPTIMIZED SOURCE CODE
504
IDENTIFY DIFFERENCES BETWEEN ORIGINAL SOURCE CODE AND OPTIMIZED SOURCE CODE
506
DISPLAY ORIGINAL SOURCE CODE AND OPTIMIZED SOURCE CODE WITH HIGHLIGHTED DIFFERENCES
508
END
510

Fig. 5

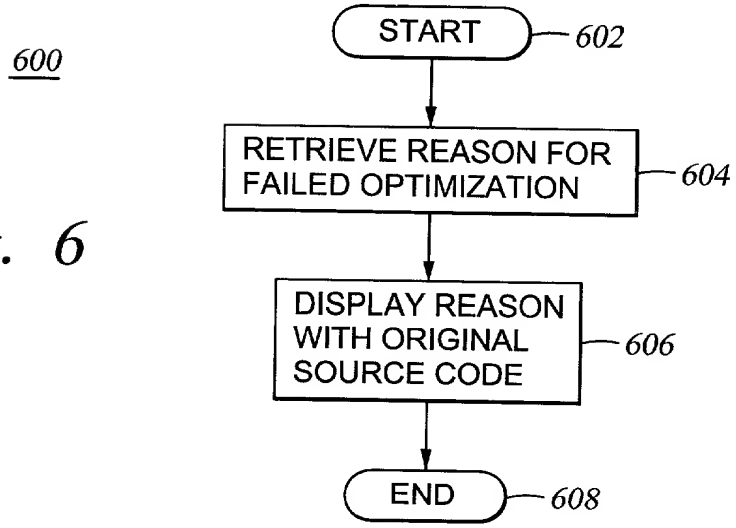

600
START
602
RETRIEVE REASON FOR FAILED OPTIMIZATION
604
DISPLAY REASON WITH ORIGINAL SOURCE CODE
606
END
608

Fig. 6